Summary of findings:

The number of mistakes Perceptron makes can be linear in total number of variables, even when number of relevant variables is kept as a small constant. The number of mistakes that Winnow makes is linear in number of relevant variables but only logarithmic in number of irrelevant variables.

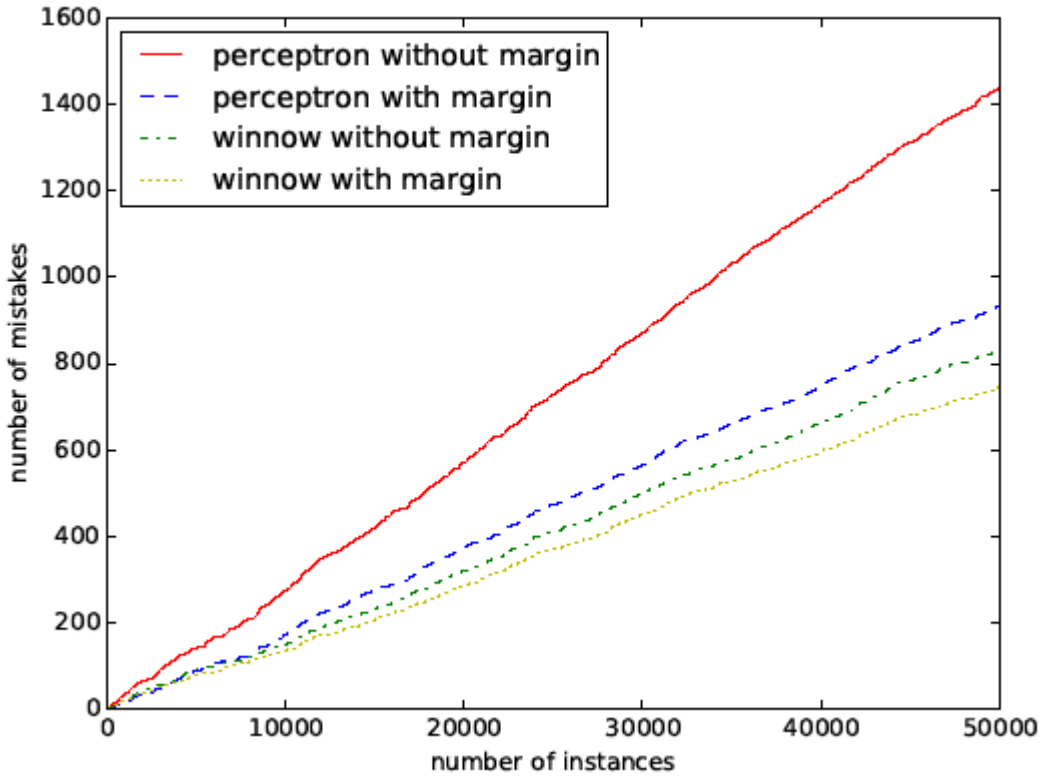
The Perceptron algorithm suffers from additional irrelevant variables much more than Winnow.

Another finding is that the Perceptron algorithm can take advantage of sparse instances: If only a few variables are active in the input at any given time but most of the variables are relevant, the Perceptron algorithm outperforms Winnow with reasonable hyper-parameter tuning. Also, Winnow in general is more sensitive to presence of noise in data.

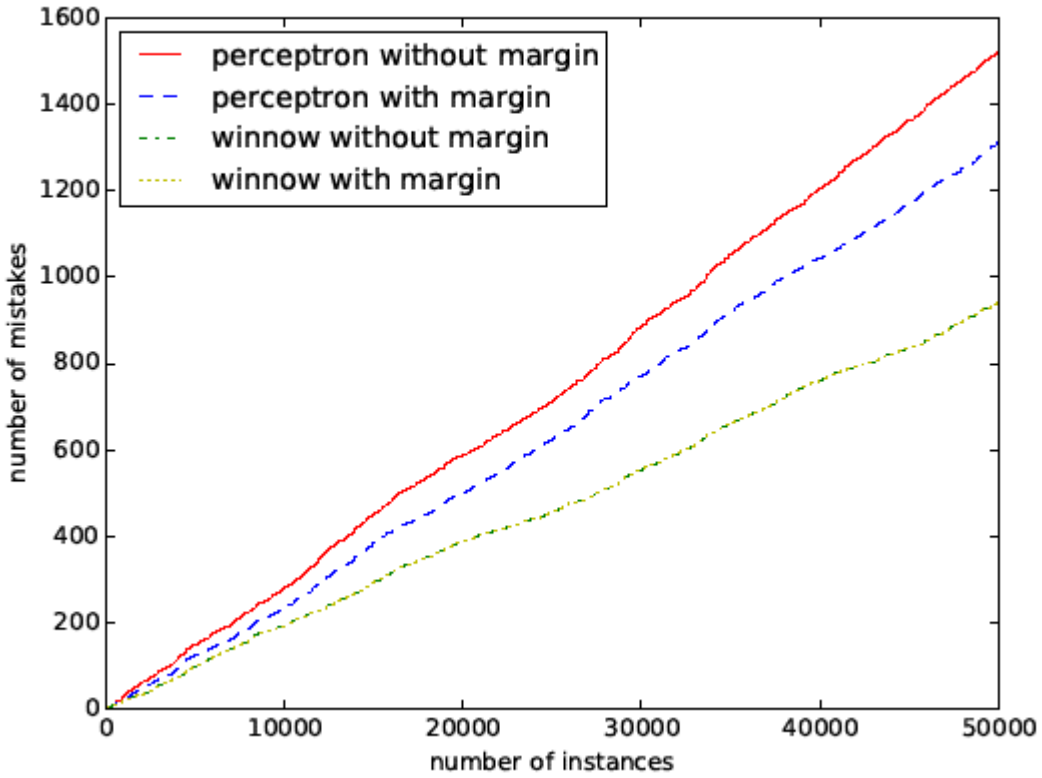
1) Table:

Algorithm	n = 500			n = 1000		
	Params	**Acc(D_2)**	**M**	**Params**	**Acc(D_2)**	**M**
Perceptron	$\gamma = 0, \eta = 1$	96.42	1423	$\gamma = 0, \eta = 1$	96.88	1575
Perceptron (γ)	$\gamma = 1, \eta = 0.001$	97.58	901	$\gamma = 1, \eta = 0.001$	97.38	1301
Winnow	$\gamma = 0, \eta = 1.1$	98.2	787	$\gamma = 0, \eta = 1.1$	98.32	871
Winnow (γ)	$\gamma = 2.0, \eta = 1.1$	98.38	697	$\gamma = 0.006, \eta = 1.1$	98.32e	871

Plots:
n = 500


perceptron without margin
perceptron with margin
winnow without margin
winnow with margin
number of mistakes
number of instances
0
200
400
600
800
1000
1200
1400
1600
0
10000
20000
30000
40000
50000

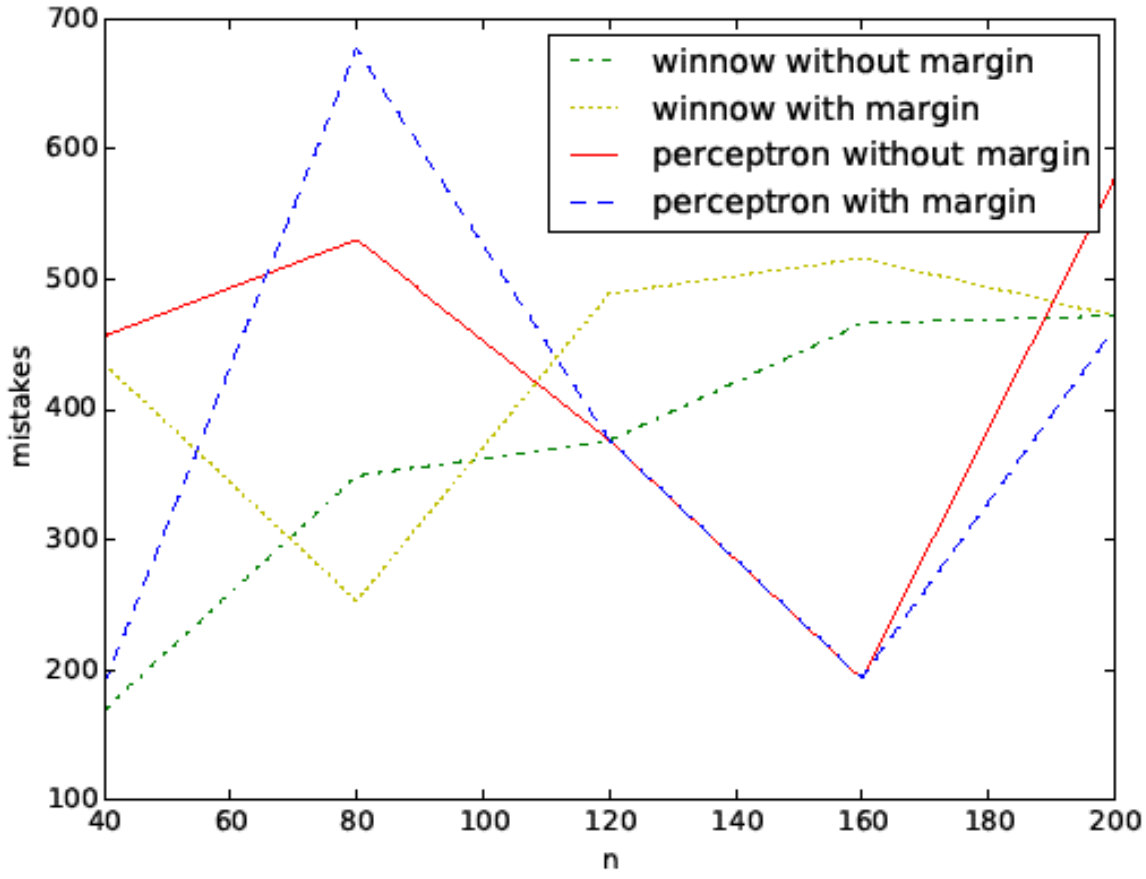
n = 1000



2) Table:

		Perceptron	Perceptron (γ)	Winnow	Winnow (γ)
n = 40	**Params**	$\gamma = 0, \eta = 1$	$\gamma = 1, \eta = 0.03$	$\gamma = 0, \eta = 1.1$	$\gamma = 0.3, \eta = 1.1$
	Acc(D_2)	98.08	99.96	99.8	99.8
	M	456	188	169	435
n = 80	**Params**	$\gamma = 0, \eta = 1$	$\gamma = 1, \eta = 0.25$	$\gamma = 0, \eta = 1.1$	$\gamma = 2.0, \eta = 1.01$
	Acc(D_2)	98.54	98.8	99.74	100
	M	530	677	349	253
n = 120	**Params**	$\gamma = 0, \eta = 1$	$\gamma = 1, \eta = 1.5$	$\gamma = 0, \eta = 1.1$	$\gamma = 2.0, \eta = 1.01$
	Acc(D_2)	97.7	98.8	99.9	100

	M	377	377	376	181
n = 160	**Params**	$\gamma = 0, \eta = 1$	$\gamma = 1, \eta = 1.5$	$\gamma = 0, \eta = 1.1$	$\gamma = 0.3, \eta = 1.1$
	Acc(D_2)	96.48	96.64	99.9	99.96
	M	194	194	466	516
n = 200	**Params**	$\gamma = 0, \eta = 1$	$\gamma = 1, \eta = 0.25$	$\gamma = 0, \eta = 1.1$	$\gamma = 2.0, \eta = 1.1$
	Acc(D_2)	95.56	97.42	99.86	99.96
	M	577	462	472	1022



As we increase the total number of variables, it has very little effect on Winnow, when the number of relevant variables is kept as a small constant. The Perceptron algorithm suffers much more from such a doubling.

3) Table:

Algorithm	m = 100			m = 500			m = 1000	
	Params	**Acc(D_2)**	**Acc(Test)**	**Params**	**Acc(D_2)**	**Acc(Test)**	**Params**	**Acc(D_2)**
Perceptron	$\gamma = 0$, $\eta = 1$	80.72	93.25	$\gamma = 0$, $\eta = 1$	83.02	93.97	$\gamma = 0$, $\eta = 1$	81.78
Perceptron ($\bar{\gamma}$)	$\gamma = 1$, $\eta = 0.001$	87.02	93.7	$\gamma = 1$, $\eta = 0.001$	88.82	94.25	$\gamma = 1$, $\eta = 0.001$	87.12
Winnow	$\gamma = 0$, $\eta = 1.0005$	81.84	94.21	$\gamma = 0$, $\eta = 1.0001$	90.58	90.81	$\gamma = 0$, $\eta = 1.005$	85.54
Winnow ($\bar{\gamma}$)	$\gamma = 0.001$, $\eta = 1.1$	84.3	94.21	$\gamma = 2.0$, $\eta = 1.0001$	90.64	90.97	$\gamma = 2.0$, $\eta = 1.005$	85.5